# From Text to Knowledge with Graphs: modelling, querying and exploiting textual content\*

Genoveva Vargas-Solar<sup>1</sup>, Mirian Halfeld Ferrari Alves<sup>2</sup>, and Anne-Lyse Minard Forst<sup>3</sup>

- CNRS, Univ Lyon, INSA Lyon, UCBL, LIRIS, UMR5205, 69622, Villeurbanne genoveva.vargas-solar@cnrs.fr
  - <sup>2</sup> University of Orléans, LIFO, Orléans mirian@univ-orleans.fr
  - <sup>3</sup> University of Orléans, LLL, Orléans anne-lyse.minard@univ-orleans.fr

**Abstract.** This paper highlights the challenges, current trends and open issues related to the representation, querying and analytics of content extracted from texts. The internet contains vast text-based information on various subjects, including commercial documents, medical records, scientific experiments, engineering tests, and events that impact urban and natural environments. Extracting knowledge from this text involves understanding the nuances of natural language and accurately representing the content without losing information. This allows knowledge to be extracted, inferred, or discovered. To achieve this, combining results from various fields, such as linguistics, natural language processing, knowledge representation, data storage, querying, and analytics, is necessary. The vision in this paper is that graphs can be a well-suited text content representation once annotated and the right querying and analytics techniques are applied. This paper discusses this hypothesis from the perspective of linguistics, natural language processing, graph models and databases and artificial intelligence provided by the panellists of the DOING session in the MADICS Symposium 2022 4.

**Keywords:** Text annotation  $\cdot$  text processing  $\cdot$  graph data models-graph analytics  $\cdot$  data science queries on graphs.

### 1 Context

Text analysis refers to the processing of texts to extract machine-readable facts. It aims to create structured data from free textual content. The process can be seen as cutting up pieces of unstructured and heterogeneous documents into manageable and interpretable data elements. From a computational point of

 $<sup>^\</sup>star$  This discussion was organised by the actions DOING and ROCED supported by the CNRS GDR MADICS of the French government.

<sup>&</sup>lt;sup>4</sup> The paper summarises the discussion in the panel of the national action DOING in the national symposium MADICS 2022 in Lyon.

view, this process requires techniques ranging from semi-automatic language processing to decode the ambiguity of human language, knowledge representation, and detection of patterns and trends representing knowledge conveyed in texts (querying in the broad sense).

The content of texts defines both a syntactic mesh and a concept mesh that can be structured as graphs. Querying these contents can be done at several levels according to the degree of abstraction this graph represents, ranging from simple guidance on the raw text to constructing a database (graph) respecting a certain number of structural constraints.

Depending on the characteristics of the graphs used, it is possible to query the content of texts in different ways: Information retrieval offers keyword methods.

- Querying graphs via query languages allows to find structural patterns and calculate aggregations.
- Machine learning and data mining methods allow to discover patterns.
- Artificial intelligence makes it possible to discover new semantic links between concepts.

Text analysis and knowledge extraction are thus addressed from different points of view, perhaps complementary, depending on the exploitation objectives through applications. This paper discusses:

[i] How text content is extracted and represented by different data models, such as matrices with term frequencies, and ontologies, by designing graph databases. [ii] The implications of the modelling choice on the possibilities of querying, updating and knowledge discovery.

The discussion is organised around a leitmotif question:

How can natural language processing, the Semantic Web, DB design for Graph DBMSs, and machine learning/data mining contribute to model/query text content?

This question has been specialised into the following research questions that have guided the analysis, statements and discussions proposed in the paper.

- RQ<sub>1</sub> What are the barriers and challenges of textual content processing? What are representative applications? Which experiments have been representative or have posed particular challenges?
- RQ<sub>2</sub> How to deal with missing information, i.e., incomplete, poor quality or contradictory annotations on property graphs and/or ontologies representing text content? What are the impacts on querying and maintenance? What issues need to be addressed?
- RQ<sub>3</sub> Maintenance and updating of text content representations in the form of property graphs or ontologies:
  - How to make annotations evolve according to the evolution of repositories, ontologies and corpora?
  - How to validate new knowledge deduced, discovered or explicitly inserted?

- RQ<sub>4</sub> What are the possibilities of querying the different representations of the text content?
  - RQ<sub>4</sub>.1 How are they exploited in NLP-based applications?
  - RQ<sub>4</sub>.2 How can querying in the "analytics" sense (data science queries with community discovery, centrality and link discovery operations) on graphs be addressed?
  - RQ<sub>4</sub>.3 What are the solutions proposed by graph-based DBMSs? How can they be positioned concerning solutions such as ML pipelines/DS pipelines?
  - RQ<sub>4</sub>.4 Is it relevant to think of a SPARQL-extended query language where the user can include a 'function' concerning graph data analysis? What are the advantages? What uses? What challenges?

The remainder of the paper proposes answers to these questions from perspectives of different disciplines: Natural Language Processing, Semantic Web, Databases and Machine learning. Existing work and open issues are identified. They all provide complementary solutions and open research perspectives to fully address the challenge of transforming textual content into knowledge to answer different questions.

Accordingly, the paper is organised as follows. Section 2 elaborates on the problems, existing solutions and open issues on how to represent textual content by exploring its syntax and semantics. Section 3 addresses text content representation challenges from the semantics point of view. It discusses how the semantic Web can contribute to representing knowledge from textual content and how it is maintained, queried and inferred. Section 4 discusses how graph database management systems can contribute to storing text content and can provide efficient methods to maintain, query and complete knowledge. Section 5 gives perspectives on how machine learning and artificial intelligence play a role in discovering knowledge within text content and from graphs when used for modelling it. Section 6 concludes the paper and wraps up answers to the questions we used to discuss about text to knowledge transformation.

# 2 Extracting knowledge from text: NLP problems and open issues

The goal of Natural Language Processing (NLP) is to allow a computer to "understand" the textual content, including the contextual nuances of the language within them [36]. NLP techniques can accurately extract text information and insights to categorise and organise the associated documents. Language-processing systems [22] have been designed by symbolic methods based on hand-coding rules and a dictionary lookup, such as writing grammar or devising heuristic rules for stemming.

Since the 90s, natural language processing research has relied on machine learning that calls for using statistical inference to learn linguistic rules through the analysis of large corpora<sup>5</sup> automatically.

<sup>&</sup>lt;sup>5</sup> Corpora refers to the plural form of a corpus, which is a set of documents, possibly with human or computer annotations.

Natural Language Processing Tasks can be classified according to their purpose:

- Text and speech processing (optical character recognition, speech recognition, speech segmentation, text-to-speech, word segmentation): the general principle is that a text in a document of some format (image, sound clip, text) determines the textual representation of the content, which can be the words/sentences composing textual content. The objective can be more or less challenging depending on the characteristics of the language in which the textual content has been produced.
- Morphological analysis (lemmatisation, morphological segmentation, part-of-speech tagging, stemming): the objective is to separate the lemmas or morphemes of the words of a text to produce normalised representations or to identify parts of speech (POS) within sentences (e.g., noun, verb, adjective).
- Syntactic analysis (grammar induction, sentence breaking, parsing) seeking to understand the structure of natural languages. The principle is to generate a grammar that describes a language's syntax or find the sentence boundaries within chunks of texts identifying punctuation marks and building parse trees (grammatical analysis) of a given sentence that represent relationships (dependencies) between words with (dependency parsing) or without probabilities (constituency parsing that produces probabilistic context-free or stochastic grammars).
- Lexical semantics of individual words in context (lexical semantics, distributional semantics, named entity recognition, sentiment analysis, terminology extraction, word-sentence disambiguation, entity linking): the objective is to determine the computational meaning of individual words in context or from data. Methods are intended to (i) identify from a stream of text which items map to proper names (e.g., people or places), and determine the type of such name (e.g. person, location, organisation); (ii) extract subjective information (e.g., sentiments); (iii) identify terminology; (iv) disambiguate words with more than one meaning.
- Relational semantics (relationship extraction, semantic parsing, semantic role labelling): identify the relationships among named entities, identify and disambiguate semantic predicates and semantic roles, and produce formal representations of the semantics of a piece of text.
- Discourse analysis goes beyond the semantics of individual sentences (conference resolution, discourse analysis, implicit semantic role labelling, recognising textual entailment, topic segmentation and recognition, argument mining). Given a sentence or a more significant chunk of text, determine which words ("mentions") refer to the same objects ("entities"). The more general task of co-reference resolution includes identifying "bridging relationships" involving referring expressions, discourse parsing for determining the nature of the discourse relationships between two sentences, recognising textual entailing, identifying topics within text segments, and identifying argumentative structures.

Textual content modelling. It is essential for NLP methods to efficiently represent syntax and semantics and exploit them to implement applications like

question and answering, automatic summarisation, grammatical error correction, and converting information from computer databases or semantic intents into readable human language, among others. The literature[16,32] recognises that NLP graphs can represent the co-occurrence of words in documents, knowledge graphs, and sentences as graphs. For every type of graph, edges can have different semantics. For example, in an ontology, nodes represent concepts and edges semantic relations; in a co-occurrence, graph nodes represent words and edges absolute or relative frequencies between two connected words or n-word window intervals, etc.

From a knowledge representation point of view, the question is how to transform textual content into graphs that fully represent the content. From a database point of view, it is a question of knowing which graph models are the most suitable for designing these graphs, e.g. property graphs, DAGs (Direct Acyclic Graphs), unDAGs (Undirected Acyclic Graphs), etc. Sections 5 and 4 will further discuss these aspects.

Text mining and NLP techniques are powerful tools for extracting structured semantic information from unstructured machine-readable text. These techniques can help uncover insights contained within text in a scalable and efficient manner by identifying and analysing explicit concepts and relationships. Some common text mining and NLP techniques include structure extraction, tokenisation, acronym normalisation, lemmatisation, decompounding, and identifying language, sentences, entities, relations, phrases, and paragraphs. These techniques help to make sense of large amounts of unstructured text data and extract valuable insights from it. As the amount and rate of text data increases, these techniques have become increasingly crucial for analysing unstructured data. Using text mining and NLP techniques, researchers can gain a deeper understanding of the world around us and make more informed decisions based on the insights extracted from large amounts of natural text data.

Many different classes of machine-learning algorithms can be applied to NLP tasks. Algorithms receive as input a large set of "features" that are generated from the input data that result from preprocessing texts. Machine learning algorithms, such as decision trees, are the basis of systems of complex if-then rules similar to existing hand-written rules. Part-of-speech tagging allows the use of hidden Markov models. NLP research has also used neural networks.

Current trends and open issues. In the field of NLP, there is a growing interest in exploring the "cognitive" aspects of natural language, as well as multilingualism and multimodality. Additionally, there are trends towards moving away from symbolic representations and evolving from rule-based to supervised, weakly supervised, and representation learning methods, as well as end-to-end systems [23].

[ $RQ_1$ ] What are the possibilities of querying the different representations of the text content? How are they exploited in NLP-based applications?

In this discussion, we are interested in the expression power of graph-based representations of the syntaxis and semantics textual content that can then be

processed to extract explicit and implicit knowledge. The focus is also on different types of querying approaches that can be applied on top of these textual content representations. For example, queries searching for factual content related to the probabilistic or effective presence of terms in textual documents as in information retrieval, aggregation queries like which is the most recurrent drogue referred to in a collection of clinical cases; inferring the protocol for medical follow-up for the treatment of disease within clinic cases described in documents; correlating a specific pattern of symptoms and the use of drogues that can deal to the observation of collateral effects reported by physicians in clinical cases.

- 1. [RQ<sub>2</sub>] What are the barriers and challenges of textual content processing for NLP methods? What are representative applications? Which experiments have been representative or have posed particular challenges?

  NLP has made significant progress in developing methods for representing and analyzing human language computationally. However, NLP methods still face several barriers and challenges when processing textual content:
  - (a) Contextual words and phrases and homonyms: Words can have different meanings depending on the context in which they are used. This can make it difficult for NLP methods to accurately interpret the intended meaning of a word or phrase.
  - (b) Synonyms: Different words can have the same meaning, which can make it difficult for NLP methods to accurately identify the intended meaning of a word or phrase.
  - (c) Irony and sarcasm: These can be difficult for NLP methods to detect and interpret accurately.
  - (d) Ambiguity: Textual content can often be ambiguous, making it difficult for NLP methods to accurately interpret the intended meaning.
  - (e) Errors in text or speech: Errors in text or speech can make it difficult for NLP methods to accurately process and interpret the content.
  - (f) Colloquialisms and slang: These can vary greatly between different regions and cultures, making it difficult for NLP methods to accurately interpret their meaning.
  - (g) Domain-specific language: Different domains can have their own specific language and terminology, which can make it difficult for NLP methods to accurately process and interpret the content.

In addition to the challenges faced by NLP methods, there is also the issue of bias. Unsupervised AI models that automatically discover hidden patterns in natural language datasets can capture linguistic regularities that reflect human biases, such as racism, sexism, and ableism <sup>6</sup>. When word embeddings are used in NLP, they can propagate bias to downstream applications, leading to biased decisions. This significant challenge must be addressed to ensure fairness and equity in using NLP methods.

 $<sup>^6</sup>$  https://www.analyticssteps.com/blogs/artificial-intelligence-healthcar e-applications-and-threats

- RQ<sub>3</sub> Maintenance and updating of text content representations: [RQ<sub>3</sub>.1] How to make annotations evolve according to the evolution of repositories? Maintaining and updating text content representations and annotations is a complex process that requires a combination of human expertise and computational methods. It is an ongoing challenge for NLP research, but it is essential for ensuring the accuracy and usefulness of NLP methods.
- RQ<sub>4</sub> What are the possibilities of querying the different representations of the text content? [RQ<sub>4</sub>.1] How can querying in the "analytics" sense (data science queries with community discovery, centrality and link discovery operations) on graphs be addressed?

Different representations of text content in NLP can be exploited in various applications to improve their performance and accuracy. For example, querying text content profiting from vectorial representations (i.e., embeddings representing words as vectors in a high-dimensional space) that capture the semantic relationships between pairs of words can allow more accurate querying of text content. Machine learning algorithms and other computational methods are used to automatically identify relevant information in text and suggest revisions or updates to improve the accuracy and consistency of the content. In NLP-based applications, these representations of text content can be exploited to improve performance and accuracy. For example, in text classification, machine learning algorithms can use word embeddings or other representations of text content to accurately classify documents into predefined categories [31]. Similarly, in information extraction, NLP methods can use these representations to identify and extract relevant information from unstructured text accurately [24,25].

NLP has many applications in healthcare, including improving clinical documentation, supporting clinical decision-making, and improving patient safety. Here are some specific applications of NLP in healthcare <sup>7</sup>:

- Clinical documentation: NLP can be used to extract information from unstructured clinical notes and convert it into structured data that can be easily analysed and used to improve patient care.
- Predictive analytics: NLP can be used to analyse electronic medical records and identify patients at greater risk of health disparities, providing an additional level of surveillance.
- Clinical decision-making: NLP can be used to support clinical decision-making by providing clinicians with relevant information from a patient's medical history, as well as the latest research and guidelines.
- Patient experience: NLP can be used to improve the patient experience by providing personalised information and support throughout the patient's journey of receiving care.

There are several challenges associated with implementing NLP in healthcare. Some of these challenges include [11]:

<sup>7</sup> https://www.mckinsey.com/industries/healthcare/our-insights/natural-lan guage-processing-in-healthcare

- Variation in language: There are many different dialects and variations of language used in medical records, which can make it difficult for NLP algorithms to accurately understand and interpret the content.
- Data standardisation: Poor standardisation of data elements, insufficient data governance policies, and variation in the design and programming of electronic health records (EHRs) can make it challenging to use NLP to fill in the gaps of structured data.
- Domain-specific language: The application of NLP methodologies for domain-specific languages, such as biomedical text, can be challenging due to the complexity and specificity of the language used.

### 3 Modelling knowledge from textual content: Semantic Web

The Semantic Web is an extension of the World Wide Web that aims to make the web more intelligent and intuitive by enabling machines to understand the meaning of information on the web. One of the main challenges in achieving this goal is dealing with textual content, which is often unstructured and difficult for machines to understand. To overcome this challenge, text mining techniques can automatically extract meaningful information and semantics from text. Another approach to dealing with textual content in the Semantic Web is the use of knowledge bases such as Wikidata <sup>8</sup>. These knowledge bases store structured, linked data that can be easily queried and visualised, making it easier for machines to understand the meaning of information on the Web. Sentiment analysis [26] is another area of research in the Semantic Web for dealing with textual content. Determining the writer's attitude towards a topic, whether positive, negative, or neutral, involves using natural language processing techniques.

Modelling textual content. Modelling knowledge from textual content with the Semantic Web involves using ontologies to represent explicit formal, conceptual models and describe semantically unstructured content and databases [42]. This approach has already seen widespread adoption in the form of Schema.org and Linked Open Data (LOD)  $^9$ .

The process of modelling text content with ontologies involves analysing the collected text for a specific domain, identifying the relevant terms, concepts and their relationships, mapping and representing the ontology by representation language (e.g. OWL (Web Ontology Language), RDF (Resource Description Framework), or RDFS (Resource Description Framework Schema)), and finally evaluating the constructed ontology [13]. The procedure of ontology construction can be done in one of three ways: manual construction, cooperative construction (need human intervention during the ontology constructing process) and (semi-) automatic construction [2].

<sup>8</sup> https://www.wikidata.org/wiki/Wikidata:Main\_Page

<sup>9</sup> https://www.ontotext.com/blog/the-semantic-web-20-years-later/

Considering the medical treatment of COVID-19, an ontology could be created by extracting key concepts and relationships from the text. For example, treatments, medications, and symptoms could be represented as concepts, and relationships such as "is used to treat," "is a symptom of," or "is a medication for" could be used to connect these concepts. Here is an example of how an ontology for medical treatment on COVID-19 might look:

Concepts: COVID-19, Antiviral Medications, Nirmatrelvir with Ritonavir (Paxlovid),

Remdesivir (Veklury), Fever, Cough, Shortness of Breath

#### Relationships:

Nirmatrelvir with Ritonavir (Paxlovid) is a medication for COVID-19 Remdesivir (Veklury) is a medication for COVID-19 Fever is a symptom of COVID-19 Cough is a symptom of COVID-19 Shortness of Breath is a symptom of COVID-19

Reasoning (querying) with ontologies. Ontologies representing text can answer various queries about the domain they represent. One type of query that can be answered using ontologies representing text is conjunctive queries, the basic database queries [8]. These are existentially quantified conjunctions of atoms, meaning they return all combinations of values that satisfy the conditions specified in the query. Conjunctive queries can retrieve information from an ontology by selecting the desired relationships between concepts. Based on the ontology above, some example queries that could be used to retrieve information from it are:

```
What are the medications for COVID-19?
Query:
SELECT ?x WHERE { ?x is a medication for COVID-19 }
Result: Nirmatrelvir with Ritonavir (Paxlovid), Remdesivir (Veklury)
What are the symptoms of COVID-19?
Query: SELECT ?x WHERE { ?x is a symptom of COVID-19 }
Result: Fever, Cough, Shortness of Breath
```

The Clinical MetaData Ontology (CMDO)  $^{10}$  is an example of an ontology representing clinical cases. It provides a simple classification scheme for elements of clinical data based on semantics and comprises 189 concepts under four first-level classes.

https://bmcmedinformdecismak.biomedcentral.com/articles/10.1186/s12911 -019-0877-x

Reasoning on text ontologies involves deriving facts not explicitly expressed in the ontology or knowledge base. Reasoning with ontologies is becoming increasingly important in data-centric applications, where it helps to integrate heterogeneous data and provide a common conceptual vocabulary.

Current trends and open issues. The following discusses the conclusions about how the combination of the Semantic Web, NLP, and AI (including ML and data mining) can be used to query textual content and extract knowledge from it. First, we address the main question of our analysis:

How can natural language processing (NLP) and the Semantic Web and machine learning/data mining contribute to model/query text content?

NLP and the Semantic Web can work together to model and query text content in several ways. For example, NLP techniques can be used to automatically extract meaningful information and semantics from text, which can then be represented using Semantic Web technologies such as RDF and OWL. This allows for the creation of structured, linked data that can be easily queried and manipulated using Semantic Web query languages such as SPARQL [33].

In addition, NLP techniques such as named entity recognition, relation extraction, and sentiment analysis can be used to enhance the querying of text content by allowing for more sophisticated queries that take into account the meaning and context of the text. For example, a query could be constructed to find all documents that mention a specific person or organisation, or to find all documents that express a positive sentiment towards a particular topic [33].

- 1. What are the barriers and challenges of textual content processing? What are representative applications? Which experiments have been representative or have posed particular challenges?
  - One of the main challenges is the difficulty of automatically extracting meaningful information and semantics from unstructured text. This process, known as ontology learning, involves using techniques from fields such as machine learning, text mining, knowledge representation and reasoning, information retrieval, and natural language processing to automatically acquire ontologies from unstructured text [50]. Another challenge is the ambiguity and complexity of natural language. Natural language labels used to describe the contents of hierarchical classifications are easily understood by human users, but can be difficult for machines to reason about due to their ambiguity [5]. This creates a challenge for converting classifications into lightweight ontologies that can be used in the Semantic Web infrastructure. In addition, building large ontologies is a difficult and time-consuming task, and it is not feasible to build ontologies for all available domains. This makes it necessary to develop automated methods for ontology learning that can scale to handle large amounts of data.
- 2. How to deal with missing information, i.e., incomplete, poor quality or contradictory annotations on property graphs and/or ontologies representing text

content? What are the impacts on querying and maintenance? What issues need to be addressed?

One approach to address this issue is to use quality assurance techniques to improve the quality of the ontology [7]. For example, one the study in [34] the detection of semantic regularities in entailed axioms can be used in ontology quality assurance, in combination with lexical techniques. Another approach is to use quality evaluation methodologies to assess the quality of the ontology and identify areas for improvement [49]. Additionally, there are tools and frameworks available that can help with the documentation and management of quality assurance measures for ontologies [46].

Incomplete data in ontologies can have a significant impact on querying and maintenance. When data is incomplete, it can be difficult to accurately answer queries or make inferences based on the available information. This can lead to incorrect or incomplete results, which can affect the usefulness and reliability of the ontology.

To address these issues, several approaches have been developed. One approach is to use ontology-based data access (OBDA) techniques to access incomplete and/or heterogeneous data [44]. These techniques allow for the use of ontologies to provide taxonomies and background knowledge that can help align and complete the data.

- 3. Maintenance and updating of text content representations in the form of property graphs or ontologies:
  - How to make annotations evolve according to the evolution of repositories, ontologies and corpora?

Ontology evolution techniques can detect changes in the ontologies and update the annotations accordingly. The work in [10] used annotators to generate more than 66 million annotations from a pre-selected set of 5000 biomedical journal articles and standard ontologies covering a period ranging from 2004 to 2016. The work highlighted the correlation between changes in the ontologies and changes in the annotations and discussed the necessity to improve existing annotation formalisms in order to include elements required to support (semi-) automatic annotation maintenance mechanisms.

Another approach is to use ontology alignment techniques to maintain the consistency of annotations across different ontologies [38,54]. It can be used to preserve the validity of ontology alignment using only the analysis of changes introduced to maintained ontologies. The precise definition of ontologies is provided, along with a definition of the ontology change log. A set of algorithms that allow revalidating ontology alignments have been built based on such elements [38].

– How to validate new knowledge deduced, discovered or explicitly inserted?

Validating new knowledge in an ontology can be done using a validation and verification driven ontology design process, such as the CQ-Driven Ontology Design Process (CODEP) proposed by Espinoza et al [17]. This process is driven by end-user requirements, defined as Competency

Questions (CQs), and includes activities that validate and verify the incremental design of an ontology through metrics based on defined CQs. Another approach is to use a method to validate the insertion of a new concept in an ontology, such as the one presented by [37]. This method is based on finding the neighbourhood of the concept in the basic ontology and assessing its semantic similarity with its neighbourhood in a general ontology, then evaluating the correlation between the values found in these steps. There are also several tools available for ontology evaluation and validation, such as OntoQA [47], which uses a set of metrics to measure different aspects of the ontology schema and knowledge base to give insight into the overall characteristics of the ontology.

- 4. What are the possibilities of querying the different representations of the text content?
  - How can we address querying in the "analytics" sense (data science queries with community discovery, centrality and link discovery operations) on graphs?

One way to query ontologies is through the use of SPARQL, a query language for RDF data. SPARQL allows users to write queries that can retrieve and manipulate data stored in RDF format. This can include data stored in triplestores, which are a type of database specifically designed for storing and querying RDF data.

Another approach to querying ontologies is through the use of reasoners, which can infer new knowledge from the existing data based on the logical rules defined in the ontology. This can allow for more complex queries that take into account the relationships between different entities and their properties.

In addition to these approaches, there are also various tools and frameworks available for querying ontologies like SPARQL, Protégé <sup>11</sup> and Apache Jena <sup>12</sup>. They can be used in conjunction with machine learning, but they do not have built-in machine learning functions.

— Is it relevant to think of a SPARQL-extended query language where the user can include a 'function' concerning graph data analysis? What are the advantages? What uses? What challenges?

SPARQL-extended query language where the user can include a 'function' concerning graph data analysis can be relevant. One example of such an extension is the addition of recursive features to the SPARQL query language [15], which allows for expressing analytical tasks over existing SPARQL infrastructure. This language is well-suited for both graph querying and analytical tasks and can express key analytical tasks on graphs [21].

The advantages of such an extension include combining querying and analytics for graphs, allowing users to perform complex analytical tasks

<sup>11</sup> https://protege.stanford.edu/

https://learn.microsoft.com/en-us/azure/architecture/data-science-process/platforms-and-tools

using a single query language. This can simplify the process of querying and analysing graph data making it easier for users to extract meaningful insights from their data [20].

Some uses of this extended query language could include performing complex graph analytics, such as finding the shortest path between two nodes, computing centrality measures, or identifying clusters or communities within the graph [35].

One challenge in developing such an extension is ensuring it is compatible with existing SPARQL infrastructure and can be easily integrated into existing systems. Another challenge is ensuring that the language is expressive enough to support a wide range of analytical tasks while remaining easy to use and understand [3].

## 4 Graph databases for storing, querying and analysing textual content

A graph database is a database management system (DBMS) that uses graph structures, including nodes, edges, and properties, to represent and store data. The fundamental concept of a graph database is the graph, which relates data items in the store to a collection of nodes and edges, with the edges representing the relationships between the nodes. Graph databases prioritise relationships between data, making querying relationships fast and efficient. Relationships can also be intuitively visualised using graph databases, making them useful for heavily inter-connected data.

Graph databases are well adapted for managing data with a graph structure, for example, the Semantic Web, networks (social, transport, biology, etc.), and financial transactions.

Graph data models. There are two popular models of graph databases: property graphs and RDF graphs. Property graphs focus on analytics and querying, while RDF graphs emphasise data integration. Both graph models consist of a collection of points (vertices) and the connections between those points (edges).

A graph data model allows to represent data in a graph structure, where data points are represented as nodes and the relationships between them are represented as edges. Graph data modelling translates a conceptual view of data into a logical model. Therefore, decision-making must determine which entities in a dataset should be nodes, which should be relationships, and which should be discarded. The result is a blueprint of data's entities, relationships, and properties.

In a graph database, nodes represent the fundamental data units, while edges represent the relationships between nodes. Properties are descriptive characteristics of nodes and edges that are not important enough to become nodes themselves. There is no formula for deriving a graph model from a dataset, but there are general guidelines that can help create an effective model. For example, it is

essential to consider the types of questions to answer with data and design the graph database accordingly  $^{13}$  [41].

The textual content representation as graphs allows to perform traversal queries based on connections, use specific algorithms on graphs, find patterns, paths, communities, influencers, single points of failure, and other relationships and have a more intuitive understanding and visualisation of the data content.

Text graphs represent a text item, such as a document, passage, or sentence, as a graph. Building a text graph is a preprocessing step in NLP to support text condensation, term disambiguation, topic-based text summarisation, relation extraction, and textual entailment. For creating a text graph, it is necessary first to identify the entities in the text that will be represented as nodes in the graph and then to identify the relationships between those entities that will be represented as edges between the nodes. The specific types of entities and relationships used will depend on the context and task. Entities can take various forms, such as individual words, bigrams, n-grams, or sequences of variable lengths. Relationships can represent a variety of connections between entities, such as adjacency in a sentence, co-occurrence within a fixed-length window, or some semantic or syntactic relationship. A directed unweighted graph (see Figure 1) can be used for representing the following text (without stop words) <sup>14</sup>:

Centrality indices are answers to the question "What characterises an important vertex?" The answer is given in terms of a real-valued function on the vertices of a graph, where the values produced are expected to provide a ranking which identifies the most important nodes. The word "importance" has a wide number of meanings, leading to many different definitions of centrality.

Text graphs are used in a variety of applications to represent and analyse textual data:

- Information retrieval: Text graphs can represent the relationships between documents and the terms they contain, allowing for more effective information retrieval.
- Text summarisation: Text graphs can represent the relationships between sentences in a document, allowing for the automatic generation of summaries.
- Topic modelling: Text graphs can represent the relationships between documents and the topics they contain, automatically identifying topics within a corpus of text.
- Sentiment analysis: Text graphs can represent the relationships between words and their sentiment, allowing for automatic sentiment analysis in text.

Querying text graphs. Regular path queries (RPQs) are a way to navigate over paths in a graph using regular expressions [12]. A regular path query (RPQ) is a regular expression  ${\bf q}$  that returns all node pairs  $({\bf u},{\bf v})$  from a graph database that

<sup>13</sup> https://cambridge-intelligence.com/graph-data-modeling-101/

This example was proposed in https://towardsdatascience.com/keyphrase-ext raction-with-graph-centrality-989e142ce427

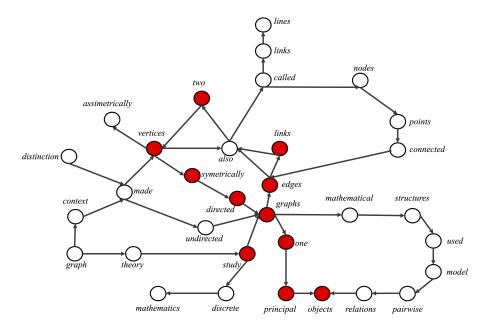


Fig. 1. Text graph example.

are connected by an arbitrary path labelled with a word from L(q). Intuitively, in an RPQ, the input is a graph, a source node and a regular expression. The goal is to identify all nodes connected to the source through a simple path whose label sequence satisfies the given regular expression. The regular expression is a formal specification of the search space that interests the user. For example, one could use an RPQ to find all paths between two nodes that have a specific node in between or to find all nodes connected by paths of a certain form  $^{15}$ .

In addition to RPQ, other types of queries can be used on text graphs:

- Attributed graph queries extract information from attributed graphs, where vertices and edges are associated with attributes such as types, numbers, and texts. For example, suppose we have an attributed graph representing a document, where each node represents a word and has an attribute "word" containing the word itself. Edges represent the order of words in the document. We can use an RPQ to find all occurrences of a specific phrase in the document. If we want to find all occurrences of the phrase "artificial intelligence" in the document, we can use the following RPQ:

[word="artificial"]/./[word="intelligence"]

- Structure extraction is used to extract the underlying structure of a text graph by identifying and grouping equivalent nodes (see Schema extraction

https://iccl.inf.tu-dresden.de/w/images/7/7f/Lecture-17-rpqs.pdf

below). For example, For example, assuming that we have a graph as described in the previous example. Suppose we want to group all occurrences of the word "artificial" together. In that case, we can use structure extraction to identify all nodes representing the word "artificial" and group them into a single node. This would result in a simplified graph where all occurrences of the word "artificial" are represented by a single node. Similarly, we can use structure extraction to group nodes representing equivalent concepts, such as synonyms or related words. We could group nodes representing the words "artificial intelligence", "AI", and "machine learning" into a single node representing the concept of artificial intelligence.

Schema inference. Frequently, graph data comes without a predefined structure and in a constraint-less fashion, thus leading to inconsistency and poor quality. Inferring the schema can allow understanding the connections in the data better, maintaining its quality and exploiting graphs through navigational and AI-based queries. Extracting a schema from property graph instances can be challenging and complex, mainly when the instances exist before the schema definition [27]. Neo4j provides a limited ability to view the schema of a database through a Cypher query that outputs a single-labelled directed graph. This graph displays the types of nodes that can be connected and the types of edges that can connect them. There are some limitations to viewing the schema of a database in Neo4j. For example, multi-labelled nodes in the graph instance are duplicated in the graph schema so that each node is assigned a single label. This results in the loss of label co-occurrence information, a critical property graph data model feature. Additionally, properties, edge cardinality constraints, and node type hierarchies are not considered. GraphQL schemas can be derived from Neo4j databases using tools such as the Neo4j Desktop GraphQL plugin or neo4j-graphql-js. These tools can infer node and edge types and the data types of node properties. However, unlike other methods, they do not infer overlapping types, node hierarchies, or nested property values. Many approaches to schema inference involve identifying structural graph summaries by grouping equivalent nodes. Clustering techniques, such as those described in [30] and [9], are commonly used to infer types in RDF datasets. These techniques are based on the assumption that the more properties two entities share, the more likely they are to belong to the same type. To achieve this, entities are grouped according to a similarity metric. Both techniques can handle hierarchical and overlapping types. In [9], a densitybased clustering method called DBSCAN is used. In contrast, [30] proposes a faster and more accurate clustering method called StaTIX, which uses the cosine similarity metric and is based on community detection. Property Graph (PG) schema inference in [27] involves inferring types, basic and complex data types, overlapping types, and node hierarchies.

Analysing text graphs. By converting free text into a graph representation, the implicit structure of the text is made explicit. This representation allows immediate access to information such as the most commonly used words (degree), the

most frequently used n-grams, and the words most commonly used to convey information (paths in the graph between every two nodes), among other things.

For example, extracting key phrases to return a list of relevant phrases (one or more words) from the input text can be possible. In this context, a relevant phrase is defined as one that is composed solely of candidate keywords, which are the most relevant unigrams. These candidate keywords are determined by graph centrality algorithms, which use the graph's structure to assign scores to nodes. In this case, the nodes represent words in the text and the relationships between them. Therefore, it is necessary to score the graph nodes (i.e., words) with a graph centrality algorithm [4,52], extract candidate keywords (i.e., most relevant individual words), extract key phrases (based on the candidate keywords).

In the previous example (see Figure 1, each node in the graph represents a term present in the input document (a single node for the multiple uses of the same term). Consider that after applying some centrality algorithm, the top 3 nodes are identified: (1) graphs: 0.56, (2) study: 0.43, (3) objects: 0.35.

For extracting the candidate keywords, we can consider a third of the total number of nodes as the number of candidate keywords. Note that the choice depends on the application context. The candidate keywords (ordered from highest to lowest scoring) are shown in figure 1:

- 1. graphs
- 2. study
- 3. objects
- 4. edges
- 5. vertices
- 6. two
- 7. link
- 8. directed
- 9. symmetrically
- 10. principal
- 11. one

A key phrase in the input document is considered relevant and can be used for various tasks, including extractive summarisation. It is defined as a sequence of words in the input document composed solely of candidate keywords. For instance, if both "mathematical" and "structures" are candidate keywords, then "mathematical structures" would be considered a key phrase, and the words would not be considered individually. In the previous text and considering the candidate keywords, examples of key phrases can be:

- 1. edges link two vertices symmetrically
- 2. directed graphs
- 3. edges link two vertices
- 4. principal objects
- 5. study
- 6. one

Current trends and open issues. One of the main advantages of using a graph store compared to graph processing libraries (e.g. NetworkX) is that graph stores are designed to handle large amounts of data and provide persistence, ACID properties, CRUD operations, commits, indexing, and other database features. This means it is possible to run graph algorithms as often as required without recreating the graph each time.

How can natural language processing, the Semantic Web, DB design for Graph DBMSs, and machine learning/data mining contribute to model/query text content?

Graph databases are focused on efficiently storing and querying highly connected data. They can use various data models for graphs and their data extensions. Some examples of graph databases include labelled property graphs [40], RDF graphs [3], and hypergraphs [6]. On the other hand, an ontology is a formal and standardised representation of knowledge used to break down data silos. It allows for defining concepts and relationships between them, emphasising class inheritance so subject matter experts can appropriately label the data in their databases. Thus, modelling text content with labelled property graphs involves representing the data as a set of nodes and edges with properties, while modelling text content with ontologies consists of defining concepts and relationships between them formally and standardised.

- 1. What are the barriers and challenges of textual content processing? What are representative applications? Which experiments have been representative or have posed particular challenges?
  - The challenges and barriers associated with textual content processing with graph databases include scalability, partitioning, processing complexity, and hardware configurations. For example, scaling graph data by distributing it in a network is much more complex than scaling simpler data models and is still a work in progress [39]. Mining complete frequent patterns from graph databases is also challenging since supporting operations are computationally costly. Also, partitioning and processing graph-structured data in parallel and distributed systems is challenging.
- 2. What are the possibilities of querying the different representations of the text content?
  - Text content can be represented as different types of graphs, and every kind of graph has specific methods for querying and retrieving information. For instance, if the text content is represented as a knowledge graph, one could use a query language such as SPARQL to retrieve information from the graph. Similarly, if the text content is represented as an RDF graph, one could use SPARQL or another RDF query language to retrieve information from the graph. If the text content is represented as a property graph, one could use a query language like Cypher or Gremlin to retrieve information from the graph.
- 3. What are the solutions proposed by graph-based DBMSs? How can they be positioned concerning solutions such as ML pipelines/DS pipelines?

Many graph databases, including Amazon Neptune, OrientDB, ArangoDB, JanusGraph, and TigerGraph, can analyse graphs using machine learning algorithms [53]. Graph-based DBMSs offer a variety of analytics and data science solutions for processing graphs. These solutions include using platforms for enterprise knowledge graphs or graph artificial intelligence (AI) and graph-based deep learning approaches to improve analytics. The graph DBMS market vendors are expanding their stacks into platforms for enterprise knowledge graphs or graph AI with associated product ecosystems. For example, graph DBMS can use a graph convolution network (GCN) to enable CRM analytics to forecast sales.

The Neo4j Graph Data Science addon provides a powerful set of tools for analysing relationships in data using graph algorithms and machine learning. The library includes a catalogue of 65+ graph algorithms, graph native ML pipelines, and graph visualisation tools, allowing users to answer questions such as what is important, what is unusual, and what is next.

Amazon Neptune has a feature called Neptune ML that uses Graph Neural Networks (GNNs) to make predictions using graph data. Neptune ML can improve the accuracy of most predictions for graphs by over 50% when compared to making predictions using non-graph methods  $^{16}$ .

OrientDB has a suggestion algorithm that leverages graph-based data modelling and machine-learning techniques to generate intelligent recommendations  $^{17}$ .

ArangoDB has a Graph Data Science Library that includes over 50 algorithms for dependencies, clustering, similarity, matching/patterns, flow, centrality, and search  $^{18}$ . Many machine learning algorithms are based on graphs. JanusGraph supports graph processing. Scaling graph data processing for real-time traversals and analytical queries is JanusGraph's foundational benefit  $^{19}$ .

TigerGraph has an in-database graph data science library that includes over 50 algorithms for dependencies, clustering, similarity, matching/patterns, flow, centrality, and search  $^{20}$ .

# 5 Beyond facts, analysing and discovering knowledge from text graphs

After extracting information from text, a significant amount of work is still required to transform that information into knowledge and valuable insights. These insights may take the form of discoveries or the confirmation and verification of previous hypotheses by creating new connections within our existing knowledge

<sup>16</sup> https://aws.amazon.com/neptune/machine-learning/

https://saturncloud.io/blog/orientdb-suggestion-algorithm-enhancing-d ata-analysis-with-intelligent-recommendations/

<sup>18</sup> https://www.arangodb.com/graph-analytics-at-enterprise-scale/

<sup>19</sup> https://docs.janusgraph.org/

<sup>20</sup> https://www.tigergraph.com/graph-data-science-library/

of a particular domain. Artificial intelligence applied to graphs is used for this purpose.

Graph analytics or Graph algorithms are analytic tools used to determine the strength and direction of relationships between objects in a graph. For example, detecting cybercrimes such as money laundering, identity fraud and cyberterrorism. Applying graph analysis to social networks and communities, such as monitoring statistics, identifying influencers and analysing the traffic and quality of service for computer networks.

Graph data can be ingested into machine learning algorithms and then be used to perform classification, clustering, regression, etc. Together, graph and machine learning provide greater analytical accuracy and faster insights. For example, graph features can work as an input for machine learning, like using PageRank score as a feature in a machine learning model. Machine learning can also enhance graph, for example, performing entity resolution <sup>21</sup> and combine different data sources into one single graph <sup>22</sup>[18,14].

A graph database and a graph data science platform can be utilized to calculate various graph metrics, including generic metrics such as betweenness centrality [19] and PageRank [43,52] score, as well as domain-specific metrics such as the number of known fraudsters indirectly connected to a given client. Additionally, these tools can be used to generate graph embeddings [48], which translate graph data into a more suitable format for machine learning.

Machine learning with Graphs is a powerful tool for addressing a wide range of problems and gain new insights into a wide range of domains. Some typical problems that can be addressed with machine learning and graphs include recommendation systems, community detection, link prediction, and node classification. Graphs can be used to represent the relationships between users and items, and machine learning algorithms can be applied to these graphs to:

- make personalised recommendations;
- identify clusters or communities of nodes that are more densely connected to each other than to the rest of the graph;
- predict the likelihood of a link forming between two nodes in a graph, based on the characteristics of the nodes and their relationships with other nodes in the graph;
- classify the nodes in a graph based on their attributes and their relationships with other nodes in the graph.

Graph Neural Networks (GNNs) are a type of neural network designed to work with graph data structures. GNNs are used in predicting nodes, edges, and graph-based tasks. The key design element of GNNs is the use of pairwise

<sup>&</sup>lt;sup>21</sup> Entity resolution is the process that resolves entities and detects relationships. The pipelines perform entity resolution as they process incoming identity records in three phases: recognise, resolve, and relate.

<sup>22</sup> https://www.ibm.com/docs/en/iii/9.0.0?topic=insight-entity-resolution

message passing, such that graph nodes iteratively update their representations by exchanging information with their neighbours. This method allows dealing with dynamic graphs [29]. GNN can also allow dealing with heterogeneous data represented as heterogeneous graphs [51]. The principle is to create view from an heterogeous graph<sup>23</sup> that contains nodes and edges of the same type and look for a vectorial representation for each type of node as shown in figure 2.

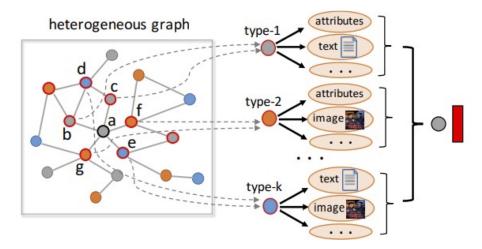


Fig. 2. Heterogeneous graph.

Vectorial representations or embeddings capture the graph topology, vertex-to-vertex relationship, and other relevant information about graphs, subgraphs, and vertices. These representations can encode each vertex with its vector representation to perform visualisation or prediction on the vertex level. For example, vertices can be visualised in a 2D plane, or new connections can be predicted based on vertex similarities.

Graph embeddings, conversely, are the transformation of property graphs to a vector or a set of vectors. These embeddings can be used to make predictions on the graph level and to compare or visualise whole graphs. For instance, graph embeddings can be used to compare chemical structures. For example, suppose we have two graphs representing social networks, and we want to compare the similarity of their community structures. We can use a graph embedding algorithm to generate embeddings for each node in both graphs. Then, we can use a similarity measure, such as cosine similarity, to compare the embeddings of corresponding nodes in the two graphs. This will give us a measure of how similar the community structures of the two graphs are.

<sup>&</sup>lt;sup>23</sup> A heterogeneous graph contains either multiple types of objects or multiple types of links. It can be used to model complex systems and relational data.

Overall, vectorial representations and embeddings are powerful tools for analyzing and understanding complex graph data. An essential challenge is a decision on the embedding dimensionality. Longer embeddings preserve more information and induce higher time and space complexity than sorter embeddings. Users need to make a trade-off based on the requirements  $^{24}$ .

Relevant application domains for GNNs include Natural Language Processing, social networks, citation networks, molecular biology, chemistry, physics and NP-hard combinatorial optimisation problems. Several open-source libraries implementing graph neural networks are available, such as PyTorch Geometric (PyTorch), TensorFlow GNN (TensorFlow), and jraph (Google JAX).

Data Science Pipelines on Text graphs. The data science pipeline involves a series of steps to gather raw data from multiple sources, analyse it, and present the results in an understandable format. This process can be applied to constructing, exploring, and analysing text graphs. General templates for these tasks can be adapted based on the algorithms and strategies used. For example, a graph construction pipeline from textual input may use information retrieval strategies to create a raw graph with relevant terms from the document represented as nodes. Rules can be established to connect the nodes based on context, such as connecting terms representing characters if separated by no more than 5 words in the text. The raw graph can then be pruned to remove dead nodes, duplicates, and isolated nodes.

To preprocess the text graph and extract knowledge from texts, the pipelines should include tasks for transforming the graphs into vectorial representations that can be used in training, testing, and evaluating GNN models. Depending on the type of graph (i.e., heterogeneous or homogeneous graph, directed or undirected, properties in the nodes and/or edges), the preprocessing pipeline can change. For example, extracting nodes and edges separately with their properties and transforming them to vectors using text2vec. In contrast, selecting graph views from a heterogeneous graph, such that the nodes and edges are the same type, and then computing a global vector representing the whole graph. Calibrating the transformation models is a recurrent process that data science pipelines can exhibit (see Figure 3 <sup>25</sup>).

Current trends and open issues.

How can data science on graphs contribute to model/query text content?

Data science on graphs can contribute to modelling and querying text content by providing a powerful way to represent and analyse the relationships between entities in a dataset. This can improve accuracy and performance for text classification, information retrieval, and natural language processing tasks.

<sup>24</sup> https://towardsdatascience.com/graph-embeddings-the-summary-cc6075aba 007

https://towardsdatascience.com/graph-embeddings-the-summary-cc6075aba 007

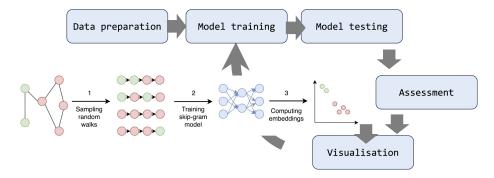


Fig. 3. Graph processing pipeline example.

- 1. What are the barriers and challenges of textual content processing? What are representative applications? Which experiments have been representative or have posed particular challenges?
  - Processing natural language involves enabling programs to produce relevant documents in good shape and automatically marking and extracting relevant information buried in the text. Preparing textual data and moving it to a target system for analytics (textual Extraction Transformation and Loading ETL) involves identifying patterns and trends in the data. Standard natural language processing approaches gain valuable knowledge in business process complexity, but this is still an active area of research. An example of a challenging application is processing the textual content in clinical cases [1]: identifying important clinical terms stated in digital text, such as treatments, disorders, and symptoms, can be challenging; locating key entities within the text: clinical text may be in an unstructured format, making it difficult to process; repeated words make it difficult to extract meaningful information.
- 2. How to deal with missing information, i.e., incomplete, poor quality or contradictory annotations on property graphs and/or ontologies representing text content? What are the impacts on querying and maintenance? What issues need to be addressed?
  - Dealing with missing information in text graph databases can be done using gradual pattern extraction techniques [45] and building an information extraction pipeline <sup>26</sup>. The pipeline involves taking unstructured text inputs, processing them with natural language processing (NLP) techniques, and using the resulting structures to populate or enrich a knowledge graph. This can help to fill in missing values and improve the overall quality of the data.
- 3. Maintenance and updating of text content representations in the form of property graphs or ontologies.
  - The approaches to making annotations evolve according to the evolution of repositories, including using an information extraction pipeline or gradual

<sup>76</sup> https://neo4j.com/blog/text-to-knowledge-graph-information-extractio n-pipeline/

pattern extraction techniques [10]. In the first case, unstructured text inputs are processed with natural language processing (NLP) techniques, and the resulting structures populate or enrich a knowledge graph. In the second case, extracting gradual patterns from property graphs provides end-users with tools for mining correlations in the data when missing values exist.

4. What are the possibilities of querying the different representations of the text content? How can we address querying in the "analytics" sense (data science queries with community discovery, centrality and link discovery operations) on graphs?

The method used to query graphs depends on the type of representation used and the user's specific needs. For instance, vector similarity search is a technique that searches for vectors based on their similarity to a given query <sup>27</sup>. This approach is commonly used in applications such as image retrieval, natural language processing, and recommendation systems. Knowledge graphs, on the other hand, integrate heterogeneous data and allow for querying large amounts of data using query languages such as SPARQL [28].

Several visual representations of text content can be queried. Concept maps are diagrams that visually represent relationships between concepts and ideas and can be used to uncover the logical structure of arguments and identify unstated assumptions. Mind maps are diagrams that visually organise information around a central idea or topic and can be searched for specific content using search and filter operations. Argument maps are visual representations of the structure of an argument, including all the key components, such as the conclusion and premises.

#### 6 Conclusions and Outlook

Graph technology forms the foundation of modern data and analytics, with capabilities to enhance and improve user collaboration, machine learning models, and explainable AI. The disciplines addressing text processing are guided by different objectives:

- 1. Content extraction: This involves using linguistic techniques to delve into the language and extract meaningful content.
- 2. Knowledge representation: This assumes that textual content defines a network of representative concepts, semantic relations, and associated consistency rules that represent the knowledge contained in the text, including knowledge that can be inferred from it.
- 3. Efficient query execution: This involves selecting appropriate graph data models to represent textual content so that specific queries can be answered efficiently.

<sup>&</sup>lt;sup>27</sup> A vector search database, also known as a vector similarity search engine or vector database, is a type of database that is designed to store, retrieve, and search for vectors based on their similarity given a query https://labelbox.com/blog/how-vector-similarity-search-works/.

4. Knowledge modelling, discovery, and prediction: Given textual content represented as graphs, this involves modelling how concepts weave content by connecting words, sentences, and groups of sentences, with the hypothesis that new knowledge can be produced and predicted. For example, graph machine learning, also known as geometric machine learning, can learn from complex data such as graphs and multi-dimensional points. Its applications have been relevant in fields such as biochemistry, drug design, and structural biology<sup>28</sup>.

The objective of analysing textual data is not only to extract and infer knowledge using artificial intelligence techniques such as machine learning, knowledge representation, and reasoning, but also to learn from the textual content and generate new knowledge. This challenge requires the integration of results to achieve previous goals. The vision is to preserve and compare the knowledge extracted from textual content, represented by graphs, within various consistency spaces. The challenge is to convey knowledge from textual content as a dynamic entity, which has already been accomplished using ontologies and graph databases, and to model its evolution and how it can be evaluated and validated within a learning process. New developments in large language models, generative AI, and reinforcement learning are propelling this ambitious trend forward. The challenges introduced necessitate the creation of new data management paradigms that take into account the characteristics of data encoding requirements (e.g., vectors and matrices) and use these encodings as first-class citizens for efficient processing. Vectorial and neural data management systems will provide persistence, indexing, and maintenance support to enable learning models to operate at scale without impedance overhead.

### Acknowledgements

The paper summarises the discussion in the panel of the national action DOING in the national symposium MADICS 2022 in Lyon. We thank the panellists who shared their insight and perspectives about the questions addressed in this paper: Donatello Conte (Polytech Tours, LIFAT), Nathalie Hernandez (Université de Toulouse, IRIT), Catherine Roussey (INRAE), Agata Savary (Université Paris-Saclay, LISN), Nicolas Travers (ELSIV, Centre de Recherche Da Vinci).

#### References

- Agrawal, S., Jain, S.K.: Medical text and image processing: applications, issues and challenges. Machine Learning with Health Care Perspective: Machine Learning and Healthcare pp. 237–262 (2020)
- 2. Al-Aswadi, F.N., Chan, H.Y., Gan, K.H.: Automatic ontology construction from text: a review from shallow to deep learning trend. Artificial Intelligence Review 53, 3901–3928 (2020)

https://www.gartner.com/smarterwithgartner/gartner-top-10-data-and-analytics-trends-for-2021

- 3. Ali, W., Saleem, M., Yao, B., Hogan, A., Ngomo, A.C.N.: A survey of rdf stores & sparql engines for querying knowledge graphs. The VLDB Journal pp. 1–26 (2022)
- 4. Arul, S.M., Senthil, G., Jayasudha, S., Alkhayyat, A., Azam, K., Elangovan, R.: Graph theory and algorithms for network analysis. In: E3S Web of Conferences. vol. 399, p. 08002. EDP Sciences (2023)
- Asim, M.N., Wasim, M., Khan, M.U.G., Mahmood, W., Abbasi, H.M.: A survey of ontology learning techniques and applications. Database 2018, bay101 (2018)
- Bellaachia, A., Al-Dhelaan, M.: Short text keyphrase extraction with hypergraphs. Progress in Artificial Intelligence 3, 73–87 (2015)
- Benbernou, S., Ouziri, M.: Enhancing data quality by cleaning inconsistent big rdf data. In: 2017 IEEE International Conference on Big Data (Big Data). pp. 74–79. IEEE (2017)
- 8. Bienvenu, M., Leclère, M., Mugnier, M.L., Rousset, M.C.: Reasoning with ontologies. A Guided Tour of Artificial Intelligence Research: Volume I: Knowledge Representation, Reasoning and Learning pp. 185–215 (2020)
- 9. Bouhamoum, R., Kellou-Menouer, K., Lopes, S., Kedad, Z.: Scaling up schema discovery for rdf datasets. In: 2018 IEEE 34th International Conference on Data Engineering Workshops (ICDEW). pp. 84–89. IEEE (2018)
- Cardoso, S.D., Pruski, C., Da Silveira, M., Lin, Y.C., Groß, A., Rahm, E., Reynaud-Delaître, C.: Leveraging the impact of ontology evolution on semantic annotations.
   In: Knowledge Engineering and Knowledge Management: 20th International Conference, EKAW 2016, Bologna, Italy, November 19-23, 2016, Proceedings 20. pp. 68–82. Springer (2016)
- Carriere, J., Shafi, H., Brehon, K., Pohar Manhas, K., Churchill, K., Ho, C., Tavakoli, M.: Case report: Utilizing ai and nlp to assist with healthcare and rehabilitation during the covid-19 pandemic. Frontiers in artificial intelligence 4, 613637 (2021)
- Chauhan, K., Jain, K., Ranu, S., Bedathur, S., Bagchi, A.: Answering regular path queries through exemplars. Proc. VLDB Endow. 15(2), 299–311 (oct 2021). https://doi.org/10.14778/3489496.3489510, https://doi.org/10.14778/3489496.3489510
- 13. Chimalakonda, S., Nori, K.V.: An ontology based modeling framework for design of educational technologies. Smart learning environments **7**(1), 1–24 (2020)
- 14. Christophides, V., Efthymiou, V., Palpanas, T., Papadakis, G., Stefanidis, K.: An overview of end-to-end entity resolution for big data. ACM Computing Surveys (CSUR) 53(6), 1–42 (2020)
- 15. Della Valle, E., Ceri, S.: Querying the semantic web: Sparql. In: Handbook of Semantic Web Technologies (2011)
- 16. Eisenstein, J.: Introduction to natural language processing. MIT press (2019)
- Espinoza, A., Del-Moral, E., Martínez-Martínez, A., Alí, N.: A validation & verification driven ontology: An iterative process. Applied Ontology 16(3), 297–337 (2021)
- Getoor, L., Machanavajjhala, A.: Entity resolution: theory, practice & open challenges. Proceedings of the VLDB Endowment 5(12), 2018–2019 (2012)
- Grando, F., Granville, L.Z., Lamb, L.C.: Machine learning in network centrality measures: Tutorial and outlook. ACM Computing Surveys (CSUR) 51(5), 1–32 (2018)
- 20. Hogan, A., Reutter, J., Soto, A.: Recursive sparql for graph analytics. arXiv preprint arXiv:2004.01816 (2020)
- 21. Hogan, A., Reutter, J.L., Soto, A.: In-database graph analytics with recursive sparql. In: International Semantic Web Conference. pp. 511–528. Springer (2020)

- 22. Idnay, B., Dreisbach, C., Weng, C., Schnall, R.: A systematic review on natural language processing systems for eligibility prescreening in clinical research. Journal of the American Medical Informatics Association **29**(1), 197–206 (2022)
- 23. Jozefowicz, R., Vinyals, O., Schuster, M., Shazeer, N., Wu, Y.: Exploring the limits of language modeling. arXiv preprint arXiv:1602.02410 (2016)
- Kaddari, Z., Mellah, Y., Berrich, J., Belkasmi, M.G., Bouchentouf, T.: Natural language processing: challenges and future directions. In: International Conference on Artificial Intelligence & Industrial Applications. pp. 236–246. Springer (2020)
- Khurana, D., Koli, A., Khatter, K., Singh, S.: Natural language processing: State
  of the art, current trends and challenges. Multimedia tools and applications 82(3),
  3713–3744 (2023)
- 26. Kouadri, W.M., Ouziri, M., Benbernou, S., Echihabi, K., Palpanas, T., Amor, I.B.: Quality of sentiment analysis tools: The reasons of inconsistency. Proceedings of the VLDB Endowment 14(4), 668–681 (2020)
- Lbath, H., Bonifati, A., Harmer, R.: Schema inference for property graphs. In: EDBT 2021-24th International Conference on Extending Database Technology. pp. 499-504 (2021)
- 28. Liang, S., Stockinger, K., de Farias, T.M., Anisimova, M., Gil, M.: Querying knowledge graphs in natural language. Journal of big data 8, 1–23 (2021)
- 29. Liu, F., Wu, J., Xue, S., Zhou, C., Yang, J., Sheng, Q.: Detecting the evolving community structure in dynamic social networks. World Wide Web **23**, 715–733 (2020)
- Lutov, A., Roshankish, S., Khayati, M., Cudré-Mauroux, P.: Statix—statistical type inference on linked data. In: 2018 IEEE International Conference on Big Data (Big Data). pp. 2253–2262. IEEE (2018)
- 31. Ma, S., Sun, X., Li, W., Li, S., Li, W., Ren, X.: Query and output: Generating words by querying distributed word representations for paraphrase generation. arXiv preprint arXiv:1803.01465 (2018)
- 32. Maulud, D.H., Zeebaree, S.R., Jacksi, K., Sadeeq, M.A.M., Sharif, K.H.: State of art for semantic analysis of natural language processing. Qubahan academic journal 1(2), 21–28 (2021)
- 33. Maynard, D., Bontcheva, K., Augenstein, I.: Natural language processing for the semantic web. Springer (2017)
- 34. Mikroyannidi, E., Quesada-Martínez, M., Tsarkov, D., Fernández Breis, J.T., Stevens, R., Palmisano, I.: A quality assurance workflow for ontologies based on semantic regularities. In: Knowledge Engineering and Knowledge Management: 19th International Conference, EKAW 2014, Linköping, Sweden, November 24-28, 2014. Proceedings 19. pp. 288–303. Springer (2014)
- Mosser, M., Pieressa, F., Reutter, J., Soto, A., Vrgoč, D.: Querying apis with sparql: language and worst-case optimal algorithms. In: The Semantic Web: 15th International Conference, ESWC 2018, Heraklion, Crete, Greece, June 3–7, 2018, Proceedings 15. pp. 639–654. Springer (2018)
- 36. Nadkarni, P.M., Ohno-Machado, L., Chapman, W.W.: Natural language processing: an introduction. Journal of the American Medical Informatics Association 18(5), 544–551 (2011)
- 37. Ngom, A.N., Diallo, P.F., Kamara-Sangaré, F., Lo, M.: A method to validate the insertion of a new concept in an ontology. In: 2016 12th International Conference on Signal-Image Technology & Internet-Based Systems (SITIS). pp. 275–281. IEEE (2016)
- 38. Pietranik, M., Kozierkiewicz, A.: Methods of managing the evolution of ontologies and their alignments. Applied Intelligence pp. 1–20 (2023)

- 39. Pokornỳ, J.: Graph databases: their power and limitations. In: Computer Information Systems and Industrial Management: 14th IFIP TC 8 International Conference, CISIM 2015, Warsaw, Poland, September 24-26, 2015, Proceedings 14. pp. 58–69. Springer (2015)
- 40. Pokornỳ, J.: Functional querying in graph databases. Vietnam journal of computer science 5(2), 95–105 (2018)
- Prevoteau, H., Djebali, S., Laiping, Z., Travers, N.: Propagation measure on circulation graphs for tourism behavior analysis. In: Proceedings of the 37th ACM/SIGAPP Symposium on Applied Computing. pp. 556–563 (2022)
- 42. Rossiello, G., Chowdhury, M.F.M., Mihindukulasooriya, N., Cornec, O., Gliozzo, A.M.: Knowgl: Knowledge generation and linking from text. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 37, pp. 16476–16478 (2023)
- 43. Sargolzaei, P., Soleymani, F.: Pagerank problem, survey and future research directions. In: International Mathematical Forum. vol. 5, pp. 937–956. Citeseer (2010)
- 44. Schneider, T., Šimkus, M.: Ontologies and data management: a brief survey. KI-Künstliche Intelligenz **34**(3), 329–353 (2020)
- 45. Shah, F., Castelltort, A., Laurent, A.: Handling missing values for mining gradual patterns from nosql graph databases. Future Generation Computer Systems 111, 523–538 (2020)
- 46. Sheveleva, T., Herrmann, K., Wawer, M.L., Kahra, C., Nürnberger, F., Koepler, O., Mozgova, I., Lachmayer, R., Auer, S.: Ontology-based documentation of quality assurance measures using the example of a visual inspection. In: International Conference on System-Integrated Intelligence. pp. 415–424. Springer (2022)
- 47. Tartir, S., Arpinar, I.B., Sheth, A.P.: Ontological evaluation and validation. Theory and applications of ontology: Computer applications pp. 115–130 (2010)
- 48. Wang, X., Bo, D., Shi, C., Fan, S., Ye, Y., Philip, S.Y.: A survey on heterogeneous graph embedding: methods, techniques, applications and sources. IEEE Transactions on Big Data 9(2), 415–436 (2022)
- 49. Wilson, R.S.I., Goonetillake, J.S., Ginige, A., Indika, W.A.: Ontology quality evaluation methodology. In: International Conference on Computational Science and Its Applications. pp. 509–528. Springer (2022)
- Zaihrayeu, I., Sun, L., Giunchiglia, F., Pan, W., Ju, Q., Chi, M., Huang, X.: From web directories to ontologies: Natural language processing challenges. In: International Semantic Web Conference. pp. 623–636. Springer (2007)
- 51. Zhang, C., Song, D., Huang, C., Swami, A., Chawla, N.V.: Heterogeneous graph neural network. In: Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining. pp. 793–803 (2019)
- Zhang, P., Wang, T., Yan, J.: Pagerank centrality and algorithms for weighted, directed networks. Physica A: Statistical Mechanics and its Applications 586, 126438 (2022)
- 53. Zhang, Z., Wang, X., Zhu, W.: Automated machine learning on graphs: A survey. arXiv preprint arXiv:2103.00742 (2021)
- 54. ZIEBELIN, M.D., PERNELLE, M.N., BROISIN, M.J., DESPRÈS, M.S., ROUS-SET, M.M.C., JOUANOT, M.F., DRUETTE, M.L.: Interactive ontology modeling and updating: Application to simulation-based training in medicine